WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
display source same display adj3 (surface or area or section)	91

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L4

Refine Search

Recall Text | Clear

Search History

DATE: Tuesday, September 23, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L4	display source same display adj3 (surface or area or section)	91	L4
L3	L2 not l1	3	L3
L2	presentation surface same display device	6	L2
L1	Primary presentation surface	3	L1

END OF SEARCH HISTORY

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L12	L10 and l5	11	L12
L11	L10 and l2	0	L11
L10	L9 same (image or object)	167	L10
L9	occlu$4 same (non-occlu$4 or nonocclu$4)	2187	L9
L8	L7 same frame	37	L8
L7	L5 same display device	706	L7
L6	L5 and l2	3	L6
L5	(image or object) adj2 (occlu$4 or view$4 or visib$5) adj6 display	3770	L5
L4	(image or object) near2 (occlu$4 or view$4 or visib$5) near6 display	15093	L4
L3	L2 and presentation surface	4	L3
L2	display source same display adj3(surface or area or section)	91	L2
L1	display source same display (surface or area or section)	63	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 11 of 11 returned.

☐ 1. Document ID: US 20030160776 A1

L12: Entry 1 of 11 File: PGPB Aug 28, 2003

DOCUMENT-IDENTIFIER: US 20030160776 A1
TITLE: Geometric folding for cone-tree data compression

Detail Description Paragraph (95):
[0125] When the search of the two trees is completed, the object attribute of each leaf-cone points to the object which is visible to the leaf-cone, and the visibility distance value of the leaf-cone specifies the distance to the visible object. This visibility information is provided to the graphics accelerator so that the graphics accelerator may render the visible objects (or visible portions of visible object) on the display device.

Detail Description Paragraph (110):
[0140] In some embodiments, each leaf hull of the hull tree may be classified as an occluder or a non-occluder. For example, a leaf hull with volume VLB which contains an object with volume VO may be classified as an occluder or non-occluder based on the magnitude of volume ratio VO/VLB. A variety of methods are contemplated for the occluder/non-occluder classification. In addition to determination (b), the processor may determine if the leaf hull is an occluder. Update operations (c) and (d) may be performed only for occluders, i.e. occluding leaf hulls. In contrast, leaf hulls that are determined to be non-occluders may be stored in a non-occluder buffer associated with the leaf cone, i.e. the cone-hull distance and object pointer associated with the leaf hull may be stored in the non-occluder buffer. Thus, the dual-tree search may identify, for each leaf cone, the N closest occluders and all non-occluders closer than the N.sup.th occluder, subject to storage limits in the non-occluder buffer(s). Upon completing the dual-tree search, the processor may transmit the nearest object pointers (i.e. the occluder pointers) and the non-occluder pointers (from the non-occluder buffer) to the rendering agent.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 2. Document ID: US 20020163515 A1

L12: Entry 2 of 11 File: PGPB Nov 7, 2002

DOCUMENT-IDENTIFIER: US 20020163515 A1
TITLE: Using ancillary geometry for visibility determination

Detail Description Paragraph (26):
[0066] When the search of the two trees is completed, the object attribute of each leaf-cone points to the object which is visible to the leaf-cone, and the visibility distance value of the leaf-cone specifies the distance to the visible object. This visibility information may be provided to a graphics accelerator so that the graphics accelerator may render the visible objects (or visible portions of visible object) on the display device.

Detail Description Paragraph (41):
[0081] In some embodiments, each object in the object collection may be classified as an occluder or a

non-occluder. For example, an object with volume VO may be classified as an occluder or non-occluder based on the magnitude of the volume ratio VO/VCH, where VCH is the volume of a convex hull containing the object. In addition, material and/or optical properties assoicated with the object may contribute a decision on the occluder or non-occluder classification. Objects made of transparent or semi-transparent materials such as glass, various liquids, plastic, fine meshes or fabrics may be classified as non-occluders. A variety of methods are contemplated for the occluder/non-occluder classification.

Detail Description Paragraph (42):
[0082] In addition to determination (b) above, the processor may determine if the leaf hull corresponds to an occluding object. Update operations (c) and (d) may be performed only for occluders. In contrast, objects that are non-occluders may be stored in a non-occluder buffer associated with the leaf cone, i.e. the ancillary distance and object pointer associated with the object may be stored in the non-occluder buffer. Thus, the dual-tree search may identify, for each leaf cone, the N closest occluders and all non-occluders closer than the N.sup.th occluder, subject to storage limits in the non-occluder buffer(s). Upon completing the dual-tree search, the processor may transmit the nearest object pointers (i.e. the occluder pointers) and the non-occluder pointers (from the non-occluder buffer) to the rendering agent.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

---

☐ 3. Document ID: US 20020050990 A1

L12: Entry 3 of 11 File: PGPB May 2, 2002

DOCUMENT-IDENTIFIER: US 20020050990 A1
TITLE: Visible-object determination for interactive visualization

Summary of Invention Paragraph (29):
[0027] When the search of the two trees is completed, the object attribute of each leaf-cone points to the object which is visible to the leaf-cone, and the visibility distance value of the leaf-cone specifies the distance to the visible object. This visibility information is provided to the graphics accelerator so that the graphics accelerator may render the visible objects (or visible portions of visible object) on the display device.

Detail Description Paragraph (160):
[0212] Non-Occluding Objects

Detail Description Paragraph (161):
[0213] Non-occluding objects are objects which do not totally occlude (i.e. block visibility) of other objects. For example, a transparent, semi-transparent, or translucent object may be a non-occluder. A screen door, tinted glass, a window with slats may be classified as non-occluders. Objects behind a non-occluder may be partially visible. The present invention contemplates certain modifications to the visibility search algorithm to allow for the presence of non-occluding objects (NOOs) in the collection of objects to be searched. In particular, the visibility search algorithm may be configured to search for the first K nearest occluding objects and any NOO closer than the K.sup.th occluder in each leaf cone, where K may be a function of leaf cone size.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

---

☐ 4. Document ID: US 20010043216 A1

L12: Entry 4 of 11 File: PGPB Nov 22, 2001

DOCUMENT-IDENTIFIER: US 20010043216 A1
TITLE: System and method for occlusion culling graphical data

Summary of Invention Paragraph (8):
[0008] Since objects can be positioned at different depths, it is possible for a first object to wholly or partially occlude a second object. This occurs when the first object and the second object have at least one set of coordinates with the same respective X and Y values but different Z values. In such a case, the first object, when displayed on the display screen, should appear to be in front of the second object. Therefore, the pixel having the foregoing X and Y values should normally be colored the appropriate color of the first object (i.e., the front object) instead of the color of the second object (i.e., the back object), assuming that the front object is not occluded by another object. As used hereafter, any object of an image frame having at least one non-occluded primitive shall be referred to as "visible."

Summary of Invention Paragraph (17):
[0016] The present invention can also be viewed as providing a graphical display method. The method can be broadly conceptualized by the following steps: receiving a graphical object of an image frame; determining, in response to the receiving step, whether the object is visible in a previously rendered image frame that is being displayed by a display device; rendering the object to a frame buffer based on the determining step; and displaying, via the display device, an image based on data stored in the frame buffer.

Detail Description Paragraph (14):
[0034] In the preferred embodiment, the system 10 implements the temporal coherency occlusion culling methodology described by copending U.S. patent application Ser. No. 09/292,906, entitled "A Method and Apparatus for Performing Occlusion Testing While Exploiting Frame-to-Frame Temporal Coherence," which is incorporated herein by reference. In temporal coherency occlusion culling, it is assumed that a high percentage of the visible objects of a previous image frame will be visible in the next successive image frame. Therefore, in rendering the next successive image frame, the objects visible in the previous frame are first rendered to a frame buffer. Then, occlusion testing is performed to determine whether any of the objects occluded in the previous frame are visible in the next frame. Any objects determined to be occluded in the previous frame but visible in the next frame are then rendered to the frame buffer. By processing the next frame such that objects previously visible in the previous frame are rendered to the frame buffer before objects previously occluded in the previous frame, it is more likely that an object occluded in the next frame will fail the occlusion test before being rendered to the frame buffer. Thus, more occluded objects fail the occlusion testing in the next frame, making the graphical display system 10 more efficient.

CLAIMS:

7. A method, comprising the steps of: receiving a graphical object of an image frame; determining, in response to said receiving step, whether said object is visible in a previously rendered image frame that is being displayed by a display device; rendering said object to a frame buffer based on said determining step; and displaying, via said display device, an image based on data stored in said frame buffer.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 5. Document ID: US 6476806 B1

L12: Entry 5 of 11 File: USPT Nov 5, 2002

DOCUMENT-IDENTIFIER: US 6476806 B1
TITLE: Method and apparatus for performing occlusion testing while exploiting frame to frame temporal coherence

CLAIMS:

11. The method of claim 8, wherein when a determination is made that a prediction error has occurred, the particular object is rendered to correct the error, and wherein the step of determining whether any rendered objects in the current image frame are occluded is accomplished by performing an occlusion test on each of the objects in the current image frame, wherein when a determination is made during the occlusion test that an object in the current image frame is occluded, the occluded object is marked with a first indicator indicating that the marked object is occluded, and wherein the step of determining whether the particular object of the current image frame was occluded in the previous image frame is performed by determining whether the object is marked with the first indicator, and wherein when a determination is made that a particular object that is not occluded in the current image frame has been marked with the first indicator, then the object is marked with a second indicator indicating that the object is visible and tagged with a tag indicating that the object is to be drawn on the display monitor, wherein each tagged object corresponds to the occurrence of a prediction error, and wherein once all of the objects of the current image frame have been occlusion tested, all of the tagged objects are rendered to the frame buffer.

30. A method for occlusion testing, comprising the steps of: identifying each object in a current image frame; determining whether each identified object was occluded in the immediately preceding image frame to the current image frame; forwarding each identified object that was not occluded in the preceding image frame to a frame buffer; determining whether each forwarded object in the frame buffer is occluded in the current image frame; determining whether each non-occluded object of the current image that was previously forwarded to the frame buffer was occluded in the preceding image frame; and generating a prediction error for each object of the current frame that was previously forwarded to the frame buffer that was occluded in the preceding image frame.

33. The method of claim 32, further comprising: marking non-occluded objects in the current image frame with a second indicator; and rendering each object of the current image frame associated with the second indicator to the frame buffer.

34. A computer program embodied on a computer-readable medium, the computer program, comprising: a code segment that identifies each object in a current image frame and determines whether each identified object in the current image frame was occluded in the immediately preceding image frame; a code segment that renders to a frame buffer the non-occluded objects identified by the code segment that identifies; a code segment that determines whether any of the objects rendered by the code segment that renders are occluded in the current image frame; a code segment, responsive to the code segment that determines, that ascertains whether non-occluded objects in the current image frame were occluded in the preceding image frame; and a code segment, responsive to the code segment that ascertains, that associates a prediction error with an object when the code segment that ascertains indicates that a non-occluded previously rendered object in the current image frame was occluded in the current image frame.

37. The program of claim 34, further comprising: a code segment, responsive to the prediction error, wherein the code segment that determines marks occluded objects with a first indicator, wherein the code segment that ascertains is responsive to the first indicator, that marks non-occluded objects with a second indicator.

☐ 6. Document ID: US 6445391 B1

L12: Entry 6 of 11 File: USPT Sep 3, 2002

DOCUMENT-IDENTIFIER: US 6445391 B1
TITLE: Visible-object determination for interactive visualization

Brief Summary Text (29):
When the search of the two trees is completed, the object attribute of each leaf-cone points to the object which is visible to the leaf-cone, and the visibility distance value of the leaf-cone specifies the distance to the visible object.

This visibility information is provided to the graphics accelerator so that the graphics accelerator may render the visible objects (or visible portions of visible object) on the display device.

Detailed Description Text (156):
Non-Occluding Objects

Detailed Description Text (157):
Non-occluding objects are objects which do not totally occlude (i.e. block visibility) of other objects. For example, a transparent, semi-transparent, or translucent object may be a non-occluder. A screen door, tinted glass, a window with slats may be classified as non-occluders. Objects behind a non-occluder may be partially visible. The present invention contemplates certain modifications to the visibility search algorithm to allow for the presence of non-occluding objects (NOOs) in the collection of objects to be searched. In particular, the visibility search algorithm may be configured to search for the first K nearest occluding objects and any NOO closer than the K.sup.th occluder in each leaf cone, where K may be a function of leaf cone size.

☐ 7. Document ID: US 6300965 B1

L12: Entry 7 of 11 File: USPT Oct 9, 2001

DOCUMENT-IDENTIFIER: US 6300965 B1
**** See image for Certificate of Correction ****
TITLE: Visible-object determination for interactive visualization

Brief Summary Text (29):
When the search of the two trees is completed, the object attribute of each leaf-cone points to the object which is visible to the leaf-cone, and the visibility distance value of the leaf-cone specifies the distance to the visible object. This visibility information is provided to the graphics accelerator so that the graphics accelerator may render the visible objects (or visible portions of visible object) on the display device.

Detailed Description Text (161):
Non-Occluding Objects

Detailed Description Text (162):
Non-occluding objects are objects which do not totally occlude (i.e. block visibility) of other objects. For example, a transparent, semi-transparent, or translucent object may be a non-occluder. A screen door, tinted glass, a window with slats may be classified as non-occluders. Objects behind a non-occluder may be partially visible. The present invention contemplates certain modifications to the visibility search algorithm to allow for the presence on non-occluding objects (NOOs) in the collection of objects to be searched. In particular, the visibility search algorithm may be configured to search for the first K nearest occluding objects and any NOO closer than the K.sup.th occluder in each leaf cone, where K may be a function of leaf cone size.

☐ 8. Document ID: US 6269565 B1

L12: Entry 8 of 11 File: USPT Aug 7, 2001

DOCUMENT-IDENTIFIER: US 6269565 B1

TITLE: Display device

Brief Summary Text (81):
A preferred edge detection algorithm for determination of the shape of an object on the viewbox display is based on the natural shape and orientation of transparencies on the viewbox display surface. Transparencies are usually rectangular. In addition, they are usually placed on the display surface so that their general orientation is either vertical or horizontal. Thus, if a non-rectangular shape appears on the viewbox, it cannot be a stand-alone transparency. It is either a foreign object or a transparency that is partially obscured by a foreign object. A preferred algorithm searches the upper strip of the viewbox for changes in intensity caused by the obstructing transparency. The horizontal extent of the transparency is determined by the extent of the obstruction. A plurality of vertical strips within that extent are analyzed to find the vertical extent of the transparency. Then the horizontal extent at the bottom of the transparency is determined. If the general determined shape is not rectangular, or if the vertical extents are not aligned to within a reasonable limit, a non-transparency object is determined to be on the viewbox display (or a combination of a transparency and a foreign object).

Brief Summary Text (91):
Other methods of determining the presence of an object occluding the viewbox according to preferred embodiments of the invention include, acquiring two images of the display surface under different local lighting conditions, such as different ambient illumination, polarization and/or backlighting, etc. and comparing the two images.

Detailed Description Text (145):
(a) acquiring at least one image of the viewbox while the display shows a preset pattern;

Detailed Description Text (235):
In preferred embodiments of the invention, in which trip-wire 530 emits light from the backlighting source, trip-wire 530 can be used to calibrate the backlighting and/or correct images for backlighting changes. For example, if the illumination, backlighting, masking or films are changed, a new reference image of viewbox 500 is preferably acquired. However, if trip-wire 530 is partially occluded, the current image cannot be used as a reference image since it not stable. Occasionally, a previously acquired image cannot be used as a reference image either, since the lighting conditions in the previous image are different from the current lighting conditions. By comparing the intensity of (the non-occluded portion of) trip-wire 530 in the current image and in the previous image, a brightness correction can be applied to the previous image so that it can be used as a reference image. In embodiments which use a uniformity map, the uniformity map can be scaled using the ratio between the trip-wire intensities in different images. Changes in the amount of ambient illumination between the previous and current images can also be determined by the brightness of the viewbox body and the intensity of the backlighting can be determined from the intensity of backlighting illuminated trip-wire 530.

Detailed Description Text (237):
Another use for trip-wire 530 is for image comparison. When two images are compared, their gray-levels are preferably normalized using their average trip-wire intensities. If part of trip-wire 530 is occluded in one of the images, only non-occluded portions of the trip-wire are used. One possible method of normalizing two images is by normalizing the images to the trip-wire intensities. Since the trip-wire is the same in both images, and the brightness histograms of the trip-wires can be aligned first, followed by the rest of the image.

Detailed Description Text (317):
The next step in the process is a step 426 of masking out the previously backlit portions of the viewbox. The new mask may be partially dimmed for general viewing of a new transparency. An additional optional step 424 of waiting may be performed, if a farewell glance is required by the viewer. The waiting period may be preset, alternatively, the farewell glance may continue until the trip-wire is not triggered or until the removal of all foreign objects from the viewbox display.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

☐ 9. Document ID: US 6111582 A

L12: Entry 9 of 11 File: USPT Aug 29, 2000

DOCUMENT-IDENTIFIER: US 6111582 A
**** See image for Certificate of Correction ****
TITLE: System and method of image generation and encoding using primitive reprojection

Brief Summary Text (61):
area produced by an object. Objects with low occluding efficiency produce little effective occlusion but result in a large emergence trail which incurs a high visibility search cost. Reprojected elements (e.g., primitives) that are covered by such objects are not considered occluded and are not removed from display lists. Classification of poorly occluding objects as non-occluding decreases the requirement for ray-cast visibility search without significantly increasing the number of elements in the display list. The net result is to decrease the computational cost of image generation.

Brief Summary Text (66):
In addition to implementation on existing distributed shared-memory multiprocessor hardware the present invention specifies an efficient 2-stage pipelined implementation in which each pipeline stage is a shared memory multiprocessor. In this embodiment corresponding subimage processors in each stage are connected using a simple message passing connection. Stage I subimage processors perform transformation, vertex lighting, and redistribution of reprojected visible object-space primitives among stage I subimage processors (using the stage I shared memory interconnect). Stage II processors receive transformed image-space primitives from the corresponding stage I subimage processor. Stage II subimage processors perform rasterization of these image-space primitives, identification of exposure regions, and ray casting or beam casting to identify newly visible primitives. Newly visible primitives identified in stage II are rasterized and subsequently communicated to stage I for later reprojective transformation. This communication employs the aforementioned interprocessor connection. Additionally the corresponding stage II processor may periodically identify occluded primitives by a read operation of the non-occluded elements in the subimage depth buffer. The array index of newly occluded primitives in the stage I primitive display list is likewise communicated to the stage I processor which effects the removal of this primitive from the primitive display list.

Drawing Description Text (44):
FIGS. 43A and 43B are illustrations showing how self-exposure regions of non-occluding objects are generated;

Detailed Description Text (10):
Specifically, FIG. 15 shows a complete primitive reprojection cycle and starts at the transformation step 1500 in which each primitive on the local display list is transformed using the concatenated transformation matrix describing object and camera motion. Control then proceeds to step 1505 which classifies to which sub-images each primitive belongs. Using this classification step 1510 determines if the primitive needs to be redistributed to other processors rendering other subimages. If check of step 1510 determines redistribution is needed, control passes to step 1525 which copies the corresponding primitive to other processors rendering subimages. (Step 1525 is depicted as being on both sides of step 1500 in order to signify that the primitive can be passed in any direction.) Whether the primitive is redistributed or not it is passed to step 1515 which also employs the classification of step 1505 to determine if the primitive is outclipped from the current subimage, and if so, control passes to step 1520 which removes the primitive from the display list of this processor. If step 1515 determines that a particular primitive is not outclipped, control passes to step 1530 which rasterizes the primitive into the depth buffer for the subimage. When all primitives in the subimage display list have been rasterized control passes to step 1540 which locates all exposure regions left in the depth buffer by the reprojection of primitives and the motion of the view frustrum. Once these exposure regions have been located control proceeds to step 1550 which performs visibility tracing in the exposure regions located in step 1540 to identify newly visible primitives. Afterwards, step 1560 adds the newly visible primitives to the local primitive reprojection list and rasterizes the newly visible primitives. The rasterization of step 1560 produces image information for the newly exposed primitives. The synthesis of the subimage data is complete on termination of step 1560 and the subimage image information can be displayed any time after step 1560. Finally, in step 1570, occluded primitives are removed from the local display list. In the present method occluded primitives are determined by a read of the depth buffer following rasterization of all previously visible primitives to identify those primitives with samples in non-occluded levels of an augmented depth buffer.

Detailed Description Text (22):
An example embodiment of the temporally double buffered depth buffer used to locate emergence regions is given

by the C data structures shown in FIG. 19. Note that the object.sub.-- class fields and "moving.sub.-- or.sub.-- not fields" as well as the multilevel structure of the buffer is employed by an extension of the method that uses object classification based on dynamic occlusive properties. Details of this object classification technique are presented in a later part of this specification. Note that the sample.sub.-- buffer.sub.-- sample data structure includes not only the current z value but a reference to the source primitive for the sample. Note also that the depth buffer is temporally double-buffered with the A and B substructures representing consecutive, alternating A and B frames of the image sequence. The present method computes visibility incrementally in the temporal domain in part by comparing the contents of frame A to that of frame B. As previously discussed emergence regions, for example, are detected by comparing the contents of frame A with frame B and identifying occlusion-nonocclusion transitions. In addition this temporal double buffering facilitates the computation of accurate exposure regions by a method of temporal supersampling described later in this specification.

Detailed Description Text (98):
In the present technique objects with a low dynamic occluding efficiency are classified as non-occluding. In this method objects classified as non-occluding do not produce exposure trails (emergence regions) in the depth buffer. By treating objects with a low dynamic occluding efficiency as non-occluding the overall size of exposure areas is decreased and the cost of visibility search decreased without significantly increasing the number of unoccluded primitives in the display list. Objects can be preclassified as non-occluding based on their shape and relationship to nearby objects. The bars of a jail cell, for example, generally have a low static occluding area from typical vantage points and usually produce relatively large exposure trails from typical view frustrum trajectories. In the present method such objects are preclassified as non-occluding and may be dynamically reclassified as occluding if their dynamic occluding efficiency changes. (As for example if the bars became very close to the viewpoint which would increase their static occluding area and possibly the dynamic occluding efficiency beyond a predetermined value.)

Detailed Description Text (99):
In the present method a multi-level depth buffer technique is employed in which primitives belonging to non-occluding objects are rasterized into the depth buffer in such a way as to not cause the overwriting or removal of ostensibly occluded samples deeper in the depth buffer. Following the rasterization of all primitives the multi-level depth buffer is read and primitives which have samples in the buffer that are actually exposed or in deeper levels of the buffer beneath non-occluding samples are indicated to be non-occluded and not removed from the display list. This is achieved in the present implementation by the aforementioned technique of writing the current frame number to the cur-frame field of the primitive C data structure shown in FIG. 28 during the aforementioned post-rasterization read of all current, non-occluded levels of the (subimage) depth buffer. In this way non-occluding objects are displayed correctly but with respect to the actual removal of occluded elements the non-occluding primitives are effectively "transparent". This increases the effective temporal visibility coherence of the image stream and reduces the cost of reprojective image generation.

Detailed Description Text (100):
An example embodiment of the multilevel depth buffer is given by the C data structures shown in FIG. 19. The "sample.sub.-- buffer.sub.-- sample" C data structure represents the information for a single sample on a single level of the buffer. It includes fields for identifying the source primitive. In this example the "element.sub.-- ID" field identifies the source primitive by giving its index in the local primitive display list. The C data structure "sample.sub.-- buffer.sub.-- element" represents data for one position in the multilevel depth buffer for an image or subimage. Each element contains two temporal sub-elements, A and B, with each temporal sub-element containing five possible levels for non-occluding samples. The C data structure Depth.sub.-- Buffer shown in FIG. 19 is a two dimensional array of these multilevel elements which comprises the entire depth buffer. Note that image information is not included in these data structures but is implemented in a separate data structure representing the top "layer" of both temporal sub-buffers. The depth buffer write and replacement protocols for resolving the visibility of occluding and non-occluding objects in the multilevel depth buffer are described in detail later in this specification.

Detailed Description Text (121):
A third method of controlling exposure gaps is to classify relatively small, rapidly moving objects as non-occluding. Such objects tend to produce large exposure gaps and, as previously discussed, can create large inefficient exposure trails. By classifying these objects as non-occluding the object does not generate an occlusion path or exposure trail. Because no emergence regions are produced by a non-occluding object exposure errors are eliminated. In the present technique an object's dynamic occluding efficiency is determined and used to classify the object as either occluding or non-occluding by the method previously described. Specific depth-comparison protocols for a multilevel depth buffer allow the visibility of both occluding and non-occluding samples to be resolved.

Detailed Description Text (122):
The aforementioned object classification system and multilevel depth comparison protocols can therefore be used not only to increase the efficiency of reprojective methods by managing objects with low dynamic occluding efficiency as non-occluding; it can also increase the accuracy of reprojective rendering by managing objects with discontinuous exposure trails as non-occluding. In addition this object classification system can also be used to allow moving objects to be rendered using reprojection; by managing them as non-occludable. Further details of the complete object classification system and multilevel depth protocols are presented following a consideration of the problems that moving objects pose for reprojective rendering.

Detailed Description Text (134):
Non-Occluding Object: Not capable of occluding other objects.

Detailed Description Text (139):
2. Non-Occluding, Occludable (NoOc)--E.g., static object with low dynamic occluding efficiency.

Detailed Description Text (141):
4. Non-occluding, Non-occludable (NoNo)--E.g., moving object with low dynamic occluding efficiency.

Detailed Description Text (143):
The four classes of objects result in 16 possible types of occlusion relationships. The occlusion relationship is determined and the depth buffer is modified for each sample according to the logic of the three flowcharts of FIGS. 39-41. In each case the occlusion relationship is established by first determining the depth order of the tested sample within the corresponding element of the sample buffer. An insertion sort of the current sample is performed with the sort details determined by the object class of the current sample and other samples in non-occluded levels of the buffer.

Detailed Description Text (144):
The depth-comparison operations of the present method serve two functions. The first function is to identify the closest visible sample for display purposes. This is the functional goal of conventional z-buffer methods. A second function of depth comparisons in the multilevel depth buffer is to identify occluded elements which can be removed from the element reprojection list. For implementations that employ sample reprojection, occluded occludable samples may be removed from the sample reprojection list. In the case of primitive reprojection primitives can be removed from the primitive reprojection list (sub-image or image) if the source object is occludable and if no samples from the primitive are exposed in the relevant (sub-image or image) region of the buffer. This is determined by the previously described read operation of the non-occluded levels of the multilevel depth buffer after all primitives have been rasterized. Thus for primitive reprojection overwriting or other removal of a sample in the multilevel depth buffer does not necessarily cause the corresponding primitive be removed from the primitive reprojection list. The depth buffer requires multiple layers so that the non-occluding, occludable objects can be represented as overlapped in the buffer. This is necessary to determine exactly which NoOc elements are occluded by any occluding object. This is determined by writing occluding samples to the buffer using a depth insertion sort such that an occluding sample will occlude samples deeper than it but will not affect non-occluding samples closer than it. The principle motivation of the multilevel depth buffer is to allow the determination of occlusion for occludable elements (e.g., primitives). In any case in which an occluding sample overwrites an occludable sample, the occludable sample, and all deeper non-occluded samples, are considered to be occluded. Subsequent read of the non-occluded levels of the buffer will establish actual occlusion of a primitive by the previously described method. Element occlusion need not be computed for each frame. Instead the frequency of determining element occlusion (e.g., primitive occlusion) can be selected to prevent the accumulation of occluded primitives in the display lists. When primitive occlusion does not need to be computed then the depth buffer write protocols revert to the simple z-buffer case. The multilevel depth buffer write protocols for determining occlusion among the four classes of objects is now described.

Detailed Description Text (145):
The case of writing samples to the multilevel depth buffer from an occluding element is described by the logic of FIG. 39. At any location of the buffer the "deepest" sample of the buffer is considered to be the occluding sample with the greatest world Z value (most distant from viewpoint). Therefore occluding samples are inserted into the corresponding element in the depth buffer in such a way as to, by definition, occlude deeper samples of any type. Occluding samples do not affect overlying (less deep) non-occluding samples at the same location of the buffer. Thus in the logic of FIG. 39, in a first step (3900) the depth order of the current sample is determined relative to other samples that have been written to the same buffer location for the current frame. If the sample is determined to be the deepest (most distant from viewpoint) of all samples at this image-space location in the buffer then control

passes to step 3910. In step 3910 the current sample is compared to the next closest (nearer to viewpoint) sample at the same image-space location. If the next closest sample is determined to be from an occluding object then control passes to 3920 and the current sample is not written to the buffer. Determination of the classification of the source object during this process is made by reading the "object.sub.-- class" field of the equivalent C data structure "sample.sub.-- buffer.sub.-- sample" shown in FIG. 19. If, on the other hand, the process of step 3910 determines that the next closer sample is from a non-occluding object then control passes to step 3930 and the current sample is written to the depth buffer at the determined depth level. In this case the sample is not occluded by the closer non-occluding sample. Returning to step 3900, if the current sample is not the deepest sample then control passes to the decision step 3940. In this step the depth of the current sample relative to the closest sample is determined. If it is determined that the current sample is the closest then control shifts to step 3950 and the current sample is written to the closest level of the buffer, overwriting the previous sample. If, on the other hand, it is determined in step 3940 that the current sample is not the closest then control shifts to step 3960 and the current sample is inserted into the list of samples in such a manner that closer non-occluding samples are unaffected. By writing an occluding sample over the non-occluding all deeper non-occluding samples are effectively overwritten. This is because on the read of the depth buffer used to establish primitive occlusion (e.g., step 1570 in FIG. 15) no samples deeper than an occluding sample are read. Note that while non-occludable samples are always overwritten by closer samples; this does not effect the determination of occlusion during the subsequent read of the non-occluded levels of the depth buffer because non-occludable primitives are explicitly exempt from occlusion. This fact makes non-occludable samples in the buffer significant only if they are occluding other (occludable) samples.

Detailed Description Text (147):
As occluding samples cause the invalidation of all occludable samples at deeper levels of the buffer the maximum number of levels required by the buffer is equal to the maximum number of overlapping Non-occluding, Occludable (NoOc) objects at any isotemporal slice (frame) of the image stream. This is generally far fewer levels than is required by the previously described Talisman architecture in which every object is allocated its own "level" in an object-based image composition scheme.

Detailed Description Text (148):
Non-Occluding, Occludable samples are written to the buffer according to the logic of FIG. 40. In a first step, 4000 the depth order of the current sample is determined relative to other samples that have been written to the same buffer location for the current frame. If the sample is determined to be the deepest (most distant from viewpoint) of all samples at this image-space location in the buffer then control passes to step 4010. In step 4010 the current sample is compared to the next closest (nearer to viewpoint) sample at the same image-space buffer element. If the next closest sample is determined to be from an occluding object then control passes to 4020 and the current sample is not written to the buffer. If, on the other hand, the process of step 4010 determines that the next closer sample is from a non-occluding object then control passes to step 4030 and the current sample is written to the depth buffer at the determined depth level. In this case the sample is not occluded by the closer non-occluding sample.

Detailed Description Text (151):
For implementations which employ primitives as the reprojected elements the preceding logic is adequate to manage the writing of non-occluding samples into the augmented depth buffer by rasterization. For implementations that employ samples as the reprojected elements the logic of FIG. 40 needs to be modified somewhat. The intrinsic image-space to object-space organization of rasterization insures that no more than one sample from each primitive occurs in a single image-space sample position. When samples are used as the reprojected elements instead of primitives, however, samples from the same primitive tend to overlap with a high depth complexity as the primitive approaches an "edge on" orientation. Retaining all of these samples in the depth buffer would require it to have a potentially large number of levels, making storage requirements for the buffer prohibitive. Therefore for implementations of the present technique which employ samples as the reprojected elements, when a non-occluding sample maps to the same location of the depth buffer as another non-occluding sample but with a deeper depth then the deeper sample is NOT written to the depth buffer. Instead the depth buffer is written with a special value in a data field equivalent to the "multiple" field of the C language data structure "sample.sub.-- buffer.sub.-- sample" in FIG. 19, indicating that more than one sample from the same Non-occluding, occludable primitive maps to this image-buffer location. Subsequent occlusion of this image-buffer location by an occluding sample would result in the removal of all of the samples that map to this location. This can be determined without an excessive number of depth levels in the buffer by searching the sample reprojection list and removing all samples with the same image-space x and y coordinates as the image buffer element that is occluded. While this is can be a relatively expensive process it often results in the removal of a substantial number of samples if the source primitive is in a near edge-on orientation. The actual search for overlapped samples in the

sample reprojection list can be limited to cases when the source primitive is in a substantially "edge-on" orientation. In one implementation of the present method the data structure equivalent to the previously described "multiple" data field for the buffer element is modified to reflect the number of samples that currently map to the corresponding position of the buffer. When this number exceeds a predetermined value, occlusion of the sample position is followed by a search in the sample reprojection list for all samples reprojecting to the same image-space location in the buffer.

Detailed Description Text (154):
In fact, implementations of the present method are substantially simplified by eliminating the NoOc classification and simply classifying all non-occluding objects as also non-occludable (NoNo). This approach substantially simplifies the depth buffer write protocols of FIGS. 39-41 by eliminating the need to perform insertion sort in the case of overlapping NoOc samples. The absence of NoOc samples also limits the number of depth levels required in a single temporal sub-layer of the augmented depth buffer to at most 2. The replacement of the NoOc class of objects with NoNo class typically results in a display list which contain somewhat more primitives (because some previously occluded primitives are instead non-occludable). This replacement results in little performance degradation, however, since the additional primitives are non-occluding and do not produce emergence trails that require visibility search. Additionally, many objects that are poorly occluding also tend to be poorly occludable in the sense that, because of their relationship with nearby potentially occluding objects, their own occlusion tends to be transient (eg. trees in a forest are both poorly occluding, ie. have a low dynamic occluding efficiency, and are poorly occludable, ie. tend to undergo transient occlusion and emergence).

Detailed Description Text (155):
NoNo objects include poorly occluding moving objects as well as static objects that are poorly occluding and poorly occludable. An example of the latter case is trees in a forest. As viewed from typical moving viewpoint trajectories (e.g., traveling along a road) trees are objects that have low occluding efficiency. Moreover from the typical viewpoint trajectory of a road, primitives comprising a model of trees in a forest are infrequently occluded by objects other than trees themselves, which generally have a low dynamic occluding efficiency from the roadway vantage point. As viewed from a moving vehicle the roadside forest has a very low temporal visibility coherence, producing a high rate of occlusion and exposure. Therefore by classifying tree primitives as non occluding and non-occludable the requirement for visibility search is dramatically reduced and the depth structure of the multilevel depth buffer is greatly simplified. Because these NoNo primitives belong to static objects (as determined by the data equivalent to the moving.sub.-- or.sub.-- not field of the C data structure primitive shown in FIG. 28) these primitives may be removed from local display lists when they outclip the corresponding subimage. This removal with subimage outclipping is possible because for the present method if such primitives actually outclip the entire viewport and are removed from all subimage display lists their subsequent reappearance in the viewport will be computed by the previously described method of visibility search (i.e., ray casting) in view volume incursion regions. Thus while these static NoNo objects do not participate in occlusion /emergence they can still be removed from display list with viewport outclipping, since they are properly identified by ray casting in view volume incursion regions if they later penetrate the view volume. This makes the performance of the present image generation method less affected by the overall size of the database than comparable image-parallel implementations of the object-order rasterization systems such as sort-first. In fact this case illustrates, once again, how the present method effectively solves the off-screen primitive problem encountered by sort-first: remove off screen primitives and find them when they come back into the viewport. In the case of trees in a forest the depth complexity of the model in the view volume incursion region can be quite high and the static occlusion coherence quite low. In this case the present method invokes the previously described technique of approximate beam tracing but without adaptive occlusion coherence search to effect a rapid search and complete search for tree primitives as they penetrate the view volume.

Detailed Description Text (160):
In addition to depth comparison protocols that are specific to object class, the present method includes object class specific techniques of depth-prioritized visibility search. In the normal execution of visibility search in exposure regions (e.g., by ray casting) the depth search is terminated when an element from an occluding (e.g., OcOc) object is encountered. Alternatively when a ray intersects an element from a non-occluding object (e.g., NoOc) the ray is further walked into the database until an occluding object is encountered or until the ray exits the database. In this way a Non-occluding object does not occlude the search for more distant newly-visible primitives behind it. In the case of visibility tracing through a non-occluding object, backfacing primitives are not excluded from ray-primitive intersections. This prevents a non-occluding object from self-occluding. Non-occluding, occludable (NoOc) and NoNo objects are not allowed to self occlude during visibility tracing because such objects do not subsequently produce an exposure trail during reprojection. This would prevent previously self-occluded elements from later being detected as newly visible in a silhouette exposure region. Because "edge-on" primitives for non-occluding objects can be missed in the initial visibility search, the special depth buffer technique for locating the self-exposure

regions of non-occluding objects is employed as previously described. Alternatively the initial detection of "edge-on" primitives can be guaranteed by including all "edge-on" primitives (determined by a surface normal substantially orthogonal to the search ray) present in every cell encountered during the ray walk.

Detailed Description Text (161):
The present method includes a special depth-buffer technique for managing a problem that can occur because non-occluding objects do not result in a self exposure trail in the buffer. In the current method non-occluding objects do produce self occlusion. During a search for newly visible elements in exposure regions, the forward facing primitives of a non-occluding convex object do not prevent the identification of backward facing primitives that would actually be obscured by the object's closer primitives. If newly visible elements (primitives or samples) are actually identified by ray-primitive intersection then "edge-on" primitives along the objects terminator are easily missed. In the present method non-occluding objects do not produce an exposure trail as previously defined by a depth buffer transition from one convex object to a more distant convex object. Without an exposure trail previously undetected "edge-on" terminator primitives would not subsequently be detected.

Detailed Description Text (162):
The present method includes a depth buffer technique by which the limited self-exposure trail caused by a non-occluding object's changing terminator can be identified. Such a trail is shown in FIG. 43B. In this technique a self-exposure trail is defined in the depth buffer for regions of the buffer that have backward facing samples from a non-occluding object without overlying forward facing samples from the same object. Such regions in the buffer define a region of potential self-exposure and are searched for previously undetected "edge-on" primitives or samples on the object's terminator. This search is conducted by using one of the depth-prioritized search techniques of the present invention. Like other emergence regions, this object's self-exposure region is susceptible to the previously described exposure errors caused by finite temporal sampling. The previously described techniques of temporal supersampling and concavity analysis can be applied to reduce the size of gaps in the object's self-exposure trail.

Detailed Description Text (163):
As previously described, the object classification system based on dynamic occlusion properties is used to determine how primitives are rasterized into an augmented multilevel depth buffer and how ray-primitive intersections are treated during visibility tracing in exposure search regions. These object classification and associated multilevel depth comparison methods can be employed to increase the accuracy, efficiency and versatility of reprojective image generation. The classification of objects has a low computational cost and can often be precomputed for specific objects based on their expected occluding properties (e.g., jail cell bars: non-occluding, walls: occluding).

Detailed Description Text (171):
into the subimage depth buffer in step 1765. Following the rasterization of the newly visible primitives the subimage is complete. Primitives identified as newly visible are passed in a retrograde fashion to stage one to be added to the local primitive reprojection list 1750. An additional step is periodically conducted in stage II following step 1765. This step is the previously described read of the depth buffer, here 1745, to identify occluded primitives. This step employs the data structures of 1745 and 1735 which is the local image-space primitive list. In this process primitives in 1735 that are not represented by any samples in non-occluded layers of the buffer are considered to be newly occluded for the subimage. The index of these primitives in the stage I (which were transmitted to stage II along with each primitive) are transmitted in a retrograde fashion to stage I where the corresponding primitives are removed from 1750.

Detailed Description Text (227):
These transformed image-space primitives are passed to a second stage where they undergo rasterization in step 6140. In this rasterization step shading is not necessarily performed since the function of rasterization in this implementation is only to write to the buffer in a way that allows exposure areas and occluded primitives to be identified in later steps. Each image-space primitive transmitted to stage II is transmitted along with its corresponding index location in the first-stage local primitive display list, 6150. Primitives transmitted to stage II from stage I processors are stored in an image-space primitive list, 6135. Following rasterization of all primitives the subimage depth buffer, 6145, is processed in step 6155 to identify exposure regions by the methods previously described. Subsequently ray casting or approximate beam casting is performed in these exposure regions, also shown as part of step 6155, in the second stage to effect the visibility search for newly exposed primitives. This process identifies newly visible primitives which are transformed and rasterized into the subimage depth buffer in step 6165. Primitives identified as newly visible are passed in a retrograde fashion to stage one to be added to the local primitive reprojection list 6150. An additional step is periodically conducted in stage II following step 6165. This step is the

previously described read of the depth buffer, here 6145, to identify occluded primitives. This step employs the data structures of 6145 and 6135 which is the local image-space primitive list. As previously described, in this process primitives in 6135 that are not represented by any samples in non-occluded layers of the buffer are considered to be newly occluded for the subimage. The index of these primitives in the stage I (which were transmitted to stage II along with each primitive) are transmitted in a retrograde fashion to stage I where the corresponding primitives are removed from 6150.

Detailed Description Text (243):
In addition to determining the list of newly visible primitives, and the list of primitives that are newly invisible because of viewport outclipping, the server computes a list of primitives that have become occluded in the image stream during the frame interval. The combination of these three sources of visibility transition represents the visibility event information for the frame and is transmitted by the server to the client. The process of determining newly occluded primitives for a frame is periodically performed by the server. This determination need not be performed for every frame. Instead the frequency of this determination is selected to prevent accumulation of occluded primitives on the client (destination) display list. This process is performed on the server by testing all newly occluded primitives for each subimage against the GVPA to determine which of the primitives newly occluded for a subimage are actually newly occluded for the entire image. The process of determining newly occluded primitives for a subimage as previously described is step 1570 of FIG. 15 for primitive reprojection. The newly occluded primitives for each subimage are determined, in the present parallel pipelined implementation, in the stage II subimage processor by the previously described process of reading non-occluded levels of the buffer after rasterization of all subimage primitives. The index of each newly occluded primitive in the stage I display list (previously transmitted with the image-space representation of the primitive) is transmitted to stage I to indicate newly occluded primitives for the subimage.

Detailed Description Text (346):
Implementations of visibility event encoding that employ primitive reprojection to generate the visibility event data stream tend to reduce transmission bandwidth requirement by the method of classifying objects with low dynamic occluding efficiency as non-occluding. Objects with low dynamic occluding efficiency (eg. trees in a forest) tend to cause transient occlusion of other primitives. This transient occlusion and the associated re-emergence tends to increases the PTR. By encoding, as newly visible, primitives that are actually occluded by objects with low dynamic occluding efficiency this type of re-encoding is largely avoided with relatively little increase in the TVL. For the method of visibility event encoding by analysis of primitive depth mattes, primitives with transient occlusion emergence behavior are identified by retrospectively analyzing the visibility event datastream produced by the analysis of consecutive primitive depth mattes. The frame packet information encoding the transient new invisibility and new visibility sequence is removed. This process causes the primitive to be retained in the DDL during its transient occlusion and eliminates inefficient transient information in the visibility event datastream.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 10. Document ID: US 6057847 A

L12: Entry 10 of 11 File: USPT May 2, 2000

DOCUMENT-IDENTIFIER: US 6057847 A
TITLE: System and method of image generation and encoding using primitive reprojection

Brief Summary Text (62):
backfacing or when they fail to clip to the view volume during the transformation clipping sequence. In addition the present method also allows the identification of primitives that remain in the view volume and are forward facing but which are completely occluded by other primitives. Such occluded primitives are also removed from the display list periodically. Efficient reprojective image generation requires that the cost of visibility search in exposure regions (e.g., by ray casting) be offset by the savings produced by the removal of occluded elements from the reprojection list. An object classification system is employed by the present invention to achieve an efficient balance between the cost of determining exposure and occlusion and the savings produced by removing occluded elements from the reprojection display list. This is achieved by classifying objects based on their occluding efficiency which is defined

as the ratio of the static occlusion area to the dynamic emergence region area produced by an object. Objects with low occluding efficiency produce little effective occlusion but result in a large emergence trail which incurs a high visibility search cost. Reprojected elements (e.g., primitives) that are covered by such objects are not considered occluded and are not removed from display lists. Classification of poorly occluding objects as non-occluding decreases the requirement for ray-cast visibility search without significantly increasing the number of elements in the display list. The net result is to decrease the computational cost of image generation.

Brief Summary Text (67):
In addition to implementation on existing distributed shared-memory multiprocessor hardware the present invention specifies an efficient 2-stage pipelined implementation in which each pipeline stage is a shared memory multiprocessor. In this embodiment corresponding subimage processors in each stage are connected using a simple message passing connection. Stage I subimage processors perform transformation, vertex lighting, and redistribution of reprojected visible object-space primitives among stage I subimage processors (using the stage I shared memory interconnect). Stage II processors receive transformed image-space primitives from the corresponding stage I subimage processor. Stage II subimage processors perform rasterization of these image-space primitives, identification of exposure regions, and ray casting or beam casting to identify newly visible primitives. Newly visible primitives identified in stage II are rasterized and subsequently communicated to stage I for later reprojective transformation. This communication employs the aforementioned interprocessor connection. Additionally the corresponding stage I processor may periodically identify occluded primitives by a read operation of the non-occluded elements in the subimage depth buffer. The array index of newly occluded primitives in the stage I primitive display list is likewise communicated to the stage I processor which effects the removal of this primitive from the primitive display list.

Drawing Description Text (44):
FIGS. 43A and 43B are illustrations showing how self-exposure regions of non-occluding objects are generated;

Detailed Description Text (10):
Specifically, FIG. 15 shows a complete primitive reprojection cycle and starts at the transformation step 1500 in which each primitive on the local display list is transformed using the concatenated transformation matrix describing object and camera motion. Control then proceeds to step 1505 which classifies to which sub-images each primitive belongs. Using this classification step 1510 determines if the primitive needs to be redistributed to other processors rendering other subimages. If check of step 1510 determines redistribution is needed, control passes to step 1525 which copies the corresponding primitive to other processors rendering subimages. (Step 1525 is depicted as being on both sides of step 1500 in order to signify that the primitive can be passed in any direction.) Whether the primitive is redistributed or not it is passed to step 1515 which also employs the classification of step 1505 to determine if the primitive is outclipped from the current subimage, and if so, control passes to step 1520 which removes the primitive from the display list of this processor. If step 1515 determines that a particular primitive is not outclipped, control passes to step 1530 which rasterizes the primitive into the depth buffer for the subimage. When all primitives in the subimage display list have been rasterized control passes to step 1540 which locates all exposure regions left in the depth buffer by the reprojection of primitives and the motion of the view frustrum. Once these exposure regions have been located control proceeds to step 1550 which performs visibility tracing in the exposure regions located in step 1540 to identify newly visible primitives. Afterwards, step 1560 adds the newly visible primitives to the local primitive reprojection list and rasterizes the newly visible primitives. The rasterization of step 1560 produces image information for the newly exposed primitives. The synthesis of the subimage data is complete on termination of step 1560 and the subimage image information can be displayed any time after step 1560. Finally, in step 1570, occluded primitives are removed from the local display list. In the present method occluded primitives are determined by a read of the depth buffer following rasterization of all previously visible primitives to identify those primitives with samples in non-occluded levels of an augmented depth buffer.

Detailed Description Text (21):
An example embodiment of the temporally double buffered depth buffer used to locate emergence regions is given by the C data structures shown in FIG. 19. Note that the object.sub.-- class fields and "moving.sub.-- or.sub.-- not fields" as well as the multilevel structure of the buffer is employed by an extension of the method that uses object classification based on dynamic occlusive properties. Details of this object classification technique are presented in a later part of this specification. Note that the sample.sub.-- buffer.sub.-- sample data structure includes not only the current z value but a reference to the source primitive for the sample. Note also that the depth buffer is temporally double-buffered with the A and B substructures representing consecutive, alternating A and B frames of the image sequence. The present method computes visibility incrementally in the temporal domain in part by comparing the contents of frame A to that of frame B. As previously discussed emergence regions, for example, are detected by comparing the contents of frame A with frame B and identifying occlusion-nonocclusion transitions. In addition this

temporal double buffering facilitates the computation of accurate exposure regions by a method of temporal supersampling described later in this specification.

Detailed Description Text (61):
In the present technique objects with a low dynamic occluding efficiency are classified as non-occluding. In this method objects classified as non-occluding do not produce exposure trails (emergence regions) in the depth buffer. By treating objects with a low dynamic occluding efficiency as non-occluding the overall size of exposure areas is decreased and the cost of visibility search decreased without significantly increasing the number of unoccluded primitives in the display list. Objects can be preclassified as non-occluding based on their shape and relationship to nearby objects. The bars of a jail cell, for example, generally have a low static occluding area from typical vantage points and usually produce relatively large exposure trails from typical view frustrum trajectories. In the present method such objects are preclassified as non-occluding and may be dynamically reclassified as occluding if their dynamic occluding efficiency changes. (As for example if the bars became very close to the viewpoint which would increase their static occluding area and possibly the dynamic occluding efficiency beyond a predetermined value.)

Detailed Description Text (62):
In the present method a multi-level depth buffer technique is employed in which primitives belonging to non-occluding objects are rasterized into the depth buffer in such a way as to not cause the overwriting or removal of ostensibly occluded samples deeper in the depth buffer. Following the rasterization of all primitives the multi-level depth buffer is read and primitives which have samples in the buffer that are actually exposed or in deeper levels of the buffer beneath non-occluding samples are indicated to be non-occluded and not removed from the display list. This is achieved in the present implementation by the aforementioned technique of writing the current frame number to the cur-frame field of the primitive C data structure shown in FIG. 28 during the aforementioned post-rasterization read of all current, non-occluded levels of the (subimage) depth buffer. In this way non-occluding objects are displayed correctly but with respect to the actual removal of occluded elements the non-occluding primitives are effectively "transparent". This increases the effective temporal visibility coherence of the image stream and reduces the cost of reprojective image generation.

Detailed Description Text (63):
An example embodiment of the multilevel depth buffer is given by the C data structures shown in FIG. 19. The "sample.sub.-- buffer.sub.-- sample" C data structure represents the information for a single sample on a single level of the buffer. It includes fields for identifying the source primitive. In this example the "element.sub.-- ID" field identifies the source primitive by giving its index in the local primitive display list. The C data structure "sample.sub.-- buffer.sub.-- element" represents data for one position in the multilevel depth buffer for an image or subimage. Each element contains two temporal sub-elements, A and B, with each temporal sub-element containing five possible levels for non-occluding samples. The C data structure Depth.sub.-- Buffer shown in FIG. 19 is a two dimensional array of these multilevel elements which comprises the entire depth buffer. Note that image information is not included in these data structures but is implemented in a separate data structure representing the top "layer" of both temporal sub-buffers. The depth buffer write and replacement protocols for resolving the visibility of occluding and non-occluding objects in the multilevel depth buffer are described in detail later in this specification.

Detailed Description Text (83):
A third method of controlling exposure gaps is to classify relatively small, rapidly moving objects as non-occluding. Such objects tend to produce large exposure gaps and, as previously discussed, can create large inefficient exposure trails. By classifying these objects as non-occluding the object does not generate an occlusion path or exposure trail. Because no emergence regions are produced by a non-occluding object exposure errors are eliminated. In the present technique an object's dynamic occluding efficiency is determined and used to classify the object as either occluding or non-occluding by the method previously described. Specific depth-comparison protocols for a multilevel depth buffer allow the visibility of both occluding and non-occluding samples to be resolved.

Detailed Description Text (84):
The aforementioned object classification system and multilevel depth comparison protocols can therefore be used not only to increase the efficiency of reprojective methods by managing objects with low dynamic occluding efficiency as non-occluding; it can also increase the accuracy of reprojective rendering by managing objects with discontinuous exposure trails as non-occluding. In addition this object classification system can also be used to allow moving objects to be rendered using reprojection; by managing them as non-occludable. Further details of the complete object classification system and multilevel depth protocols are presented following a consideration of the problems that moving objects pose for reprojective rendering. Graphic elements (primitives or samples) that belong to objects which are moving in three-space can become exposed anywhere in the image stream. Unlike

static objects their exposure is not limited to the previously described emergence regions and view volume incursion regions. This inability to efficiently compute the visibility of moving objects is a major limitation of existing methods of reprojective image generation.

Detailed Description Text (95):
Non-Occluding Object: Not capable of occluding other objects.

Detailed Description Text (100):
2. Non-Occluding, Occludable (NoOc)--E.g., static object with low dynamic occluding efficiency.

Detailed Description Text (102):
4. Non-occluding, Non-occludable (NoNo)--E.g., moving object with low dynamic occluding efficiency.

Detailed Description Text (104):
The four classes of objects result in 16 possible types of occlusion relationships. The occlusion relationship is determined and the depth buffer is modified for each sample according to the logic of the three flowcharts of FIGS. 39-41. In each case the occlusion relationship is established by first determining the depth order of the tested sample within the corresponding element of the sample buffer. An insertion sort of the current sample is performed with the sort details determined by the object class of the current sample and other samples in non-occluded levels of the buffer.

Detailed Description Text (105):
The depth-comparison operations of the present method serve two functions. The first function is to identify the closest visible sample for display purposes. This is the functional goal of conventional z-buffer methods. A second function of depth comparisons in the multilevel depth buffer is to identify occluded elements which can be removed from the element reprojection list. For implementations that employ sample reprojection, occluded occludable samples may be removed from the sample reprojection list. In the case of primitive reprojection primitives can be removed from the primitive reprojection list (sub-image or image) if the source object is occludable and if no samples from the primitive are exposed in the relevant (sub-image or image) region of the buffer. This is determined by the previously described read operation of the non-occluded levels of the multilevel depth buffer after all primitives have been rasterized. Thus for primitive reprojection overwriting or other removal of a sample in the multilevel depth buffer does not necessarily cause the corresponding primitive be removed from the primitive reprojection list. The depth buffer requires multiple layers so that the non-occluding, occludable objects can be represented as overlapped in the buffer. This is necessary to determine exactly which NoOc elements are occluded by any occluding object. This is determined by writing occluding samples to the buffer using a depth insertion sort such that an occluding sample will occlude samples deeper than it but will not affect non-occluding samples closer than it. The principle motivation of the multilevel depth buffer is to allow the determination of occlusion for occludable elements (e.g., primitives). In any case in which an occluding sample overwrites an occludable sample, the occludable sample, and all deeper non-occluded samples, are considered to be occluded. Subsequent read of the non-occluded levels of the buffer will establish actual occlusion of a primitive by the previously described method. Element occlusion need not be computed for each frame. Instead the frequency of determining element occlusion (e.g., primitive occlusion) can be selected to prevent the accumulation of occluded primitives in the display lists. When primitive occlusion does not need to be computed then the depth buffer write protocols revert to the simple z-buffer case. The multilevel depth buffer write protocols for determining occlusion among the four classes of objects is now described.

Detailed Description Text (106):
The case of writing samples to the multilevel depth buffer from an occluding element is described by the logic of FIG. 39. At any location of the buffer the "deepest" sample of the buffer is considered to be the occluding sample with the greatest world Z value (most distant from viewpoint). Therefore occluding samples are inserted into the corresponding element in the depth buffer in such a way as to, by definition, occlude deeper samples of any type. Occluding samples do not affect overlying (less deep) non-occluding samples at the same location of the buffer. Thus in the logic of FIG. 39, in a first step (3900) the depth order of the current sample is determined relative to other samples that have been written to the same buffer location for the current frame. If the sample is determined to be the deepest (most distant from viewpoint) of all samples at this image-space location in the buffer then control passes to step 3910. In step 3910 the current sample is compared to the next closest (nearer to viewpoint) sample at the same image-space location. If the next closest sample is determined to be from an occluding object then control passes to 3920 and the current sample is not written to the buffer. Determination of the classification of the source object during this process is made by reading the "object.sub.-- class" field of the equivalent C data structure "sample.sub.-- buffer.sub.-- sample" shown in FIG. 19. If, on the other hand, the process of step 3910 determines

that the next closer sample is from a non-occluding object then control passes to step 3930 and the current sample is written to the depth buffer at the determined depth level. In this case the sample is not occluded by the closer non-occluding sample. Returning to step 3900, if the current sample is not the deepest sample then control passes to the decision step 3940. In this step the depth of the current sample relative to the closest sample is determined. If it is determined that the current sample is the closest then control shifts to step 3950 and the current sample is written to the closest level of the buffer, overwriting the previous sample. If, on the other hand, it is determined in step 3940 that the current sample is not the closest then control shifts to step 3960 and the current sample is inserted into the list of samples in such a manner that closer non-occluding samples are unaffected. By writing an occluding sample over the non-occluding all deeper non-occluding samples are effectively overwritten. This is because on the read of the depth buffer used to establish primitive occlusion (e.g., step 1570 in FIG. 15) no samples deeper than an occluding sample are read. Note that while non-occludable samples are always overwritten by closer samples; this does not effect the determination of occlusion during the subsequent read of the non-occluded levels of the depth buffer because non-occludable primitives are explicitly exempt from occlusion. This fact makes non-occludable samples in the buffer significant only if they are occluding other (occludable) samples.

Detailed Description Text (108):
As occluding samples cause the invalidation of all occludable samples at deeper levels of the buffer the maximum number of levels required by the buffer is equal to the maximum number of overlapping Non-occluding, Occludable (NoOc) objects at any isotemporal slice (frame) of the image stream. This is generally far fewer levels than is required by the previously described Talisman architecture in which every object is allocated its own "level" in an object-based image composition scheme.

Detailed Description Text (109):
Non-Occluding, Occludable samples are written to the buffer according to the logic of FIG. 40. In a first step, 4000 the depth order of the current sample is determined relative to other samples that have been written to the same buffer location for the current frame. If the sample is determined to be the deepest (most distant from viewpoint) of all samples at this image-space location in the buffer then control passes to step 4010. In step 4010 the current sample is compared to the next closest (nearer to viewpoint) sample at the same image-space buffer element. If the next closest sample is determined to be from an occluding object then control passes to 4020 and the current sample is not written to the buffer. If, on the other hand, the process of step 4010 determines that the next closer sample is from a non-occluding object then control passes to step 4030 and the current sample is written to the depth buffer at the determined depth level. In this case the sample is not occluded by the closer non-occluding sample.

Detailed Description Text (112):
For implementations which employ primitives as the reprojected elements the preceding logic is adequate to manage the writing of non-occluding samples into the augmented depth buffer by rasterization. For implementations that employ samples as the reprojected elements the logic of FIG. 40 needs to be modified somewhat. The intrinsic image-space to object-space organization of rasterization insures that no more than one sample from each primitive occurs in a single image-space sample position. When samples are used as the reprojected elements instead of primitives, however, samples from the same primitive tend to overlap with a high depth complexity as the primitive approaches an "edge on" orientation. Retaining all of these samples in the depth buffer would require it to have a potentially large number of levels, making storage requirements for the buffer prohibitive. Therefore for implementations of the present technique which employ samples as the reprojected elements, when a non-occluding sample maps to the same location of the depth buffer as another non-occluding sample but with a deeper depth then the deeper sample is NOT written to the depth buffer. Instead the depth buffer is written with a special value in a data field equivalent to the "multiple" field of the C language data structure "sample.sub.-- buffer.sub.-- sample" in FIG. 19, indicating that more than one sample from the same Non-occluding, occludable primitive maps to this image-buffer location. Subsequent occlusion of this image-buffer location by an occluding sample would result in the removal of all of the samples that map to this location. This can be determined without an excessive number of depth levels in the buffer by searching the sample reprojection list and removing all samples with the same image-space x and y coordinates as the image buffer element that is occluded. While this is can be a relatively expensive process it often results in the removal of a substantial number of samples if the source primitive is in a near edge-on orientation. The actual search for overlapped samples in the sample reprojection list can be limited to cases when the source primitive is in a substantially "edge-on" orientation. In one implementation of the present method the data structure equivalent to the previously described "multiple" data field for the buffer element is modified to reflect the number of samples that currently map to the corresponding position of the buffer. When this number exceeds a predetermined value, occlusion of the sample position is followed by a search in the sample reprojection list for all samples reprojecting to the same image-space

location in the buffer.

Detailed Description Text (113):
A Non-occluding, Non-Occludable (NoNo) sample is written to the multilevel depth buffer only if it is the current closest sample in the corresponding buffer element. Since such samples neither occlude nor can be occluded they effectively do not participate in the reprojective visibility solution. In this respect the multilevel depth buffer functions as a conventional z-buffer for NoNo samples. The logic of writing NoNo samples to the depth buffer is shown in FIG. 41. This fact can be exploited when the number of overlapping NoOc samples exceeds the number of levels in the buffer by reclassifying some of the NoOc objects as NoNo thereby reducing the number of levels required.

Detailed Description Text (115):
be removed from local display lists when they outclip the corresponding subimage. This removal with subimage outclipping is possible because for the present method if such primitives actually outclip the entire viewport and are removed from all subimage display lists their subsequent reappearance in the viewport will be computed by the previously described method of visibility search (i.e., ray casting) in view volume incursion regions. Thus while these static NoNo objects do not participate in occlusion/emergence they can still be removed from display list with viewport outclipping, since they are properly identified by ray casting in view volume incursion regions if they later penetrate the view volume. This makes the performance of the present image generation method less affected by the overall size of the database than comparable image-parallel implementations of the object-order rasterization systems such as sort-first. In fact this case illustrates, once again, how the present method effectively solves the off-screen primitive problem encountered by sort-first: remove off screen primitives and find them when they come back into the viewport. In the case of trees in a forest the depth complexity of the model in the view volume incursion region can be quite high and the static occlusion coherence quite low. In this case the present method invokes the previously described technique of approximate beam tracing but without adaptive occlusion coherence search to effect a rapid search and complete search for tree primitives as they penetrate the view volume.

Detailed Description Text (120):
In addition to depth comparison protocols that are specific to object class, the present method includes object class specific techniques of depth-prioritized visibility search. In the normal execution of visibility search in exposure regions (e.g., by ray casting) the depth search is terminated when an element from an occluding (e.g., OcOc) object is encountered. Alternatively when a ray intersects an element from a non-occluding object (e.g., NoOc) the ray is further walked into the database until an occluding object is encountered or until the ray exits the database. In this way a Non-occluding object does not occlude the search for more distant newly-visible primitives behind it. In the case of visibility tracing through a non-occluding object, backfacing primitives are not excluded from ray-primitive intersections. This prevents a non-occluding object from self-occluding. Non-occluding, occludable (NoOc) and NoNo objects are not allowed to self occlude during visibility tracing because such objects do not subsequently produce an exposure trail during reprojection. This would prevent previously self-occluded elements from later being detected as newly visible in a silhouette exposure region. Because "edge-on" primitives for non-occluding objects can be missed in the initial visibility search, the special depth buffer technique for locating the self-exposure regions of non-occluding objects is employed as previously described. Alternatively the initial detection of "edge-on" primitives can be guaranteed by including all "edge-on" primitives (determined by a surface normal substantially orthogonal to the search ray) present in every cell encountered during the ray walk.

Detailed Description Text (121):
The present method includes a special depth-buffer technique for managing a problem that can occur because non-occluding objects do not result in a self exposure trail in the buffer. In the current method non-occluding objects do produce self occlusion. During a search for newly visible elements in exposure regions, the forward facing primitives of a non-occluding convex object do not prevent the identification of backward facing primitives that would actually be obscured by the object's closer primitives. If newly visible elements (primitives or samples) are actually identified by ray-primitive intersection then "edge-on" primitives along the objects terminator are easily missed. In the present method non-occluding objects do not produce an exposure trail as previously defined by a depth buffer transition from one convex object to a more distant convex object. Without an exposure trail previously undetected "edge-on" terminator primitives would not subsequently be detected.

Detailed Description Text (122):
The present method includes a depth buffer technique by which the limited self-exposure trail caused by a non-occluding object's changing terminator can be identified. Such a trail is shown in FIG. 43B. In this technique a self-exposure trail is defined in the depth buffer for regions of the buffer that have backward facing samples from a non-occluding object without overlying forward facing samples from the same object. Such regions in the buffer define a region of potential self-exposure and are searched for previously undetected "edge-on" primitives or

samples on the object's terminator. This search is conducted by using one of the depth-prioritized search techniques of the present invention. Like other emergence regions, this object's self-exposure region is susceptible to the previously described exposure errors caused by finite temporal sampling. The previously described techniques of temporal supersampling and concavity analysis can be applied to reduce the size of gaps in the object's self-exposure trail.

Detailed Description Text (123):
As previously described, the object classification system based on dynamic occlusion properties is used to determine how primitives are rasterized into an augmented multilevel depth buffer and how ray-primitive intersections are treated during visibility tracing in exposure search regions. These object classification and associated multilevel depth comparison methods can be employed to increase the accuracy, efficiency and versatility of reprojective image generation. The classification of objects has a low computational cost and can often be precomputed for specific objects based on their expected occluding properties (e.g., jail cell bars: non-occluding, walls: occluding).

Detailed Description Text (130):
These transformed image-space primitives are passed to a second stage where they undergo rasterization in step 1740. Each image-space primitive transmitted to stage II is transmitted along with its corresponding index location in the first-stage local primitive display list, 1750. Primitives transmitted to stage II from stage I processors are stored in an image-space primitive list, 1735. Following rasterization of all primitives, the subimage depth buffer, 1745, is processed in step 1755 to identify exposure regions by the methods previously described. Subsequently ray casting or beam casting is performed in these exposure regions, also shown as part of step 1755, in the second stage to effect the visibility search for newly exposed primitives. This process identifies newly visible primitives which are transformed and rasterized into the subimage depth buffer in step 1765. Following the rasterization of the newly visible primitives the subimage is complete. Primitives identified as newly visible are passed in a retrograde fashion to stage one to be added to the local primitive reprojection list 1750. An additional step is periodically conducted in stage II following step 1765. This step is the previously described read of the depth buffer, here 1745, to identify occluded primitives. This step employs the data structures of 1745 and 1735 which is the local image-space primitive list. In this process primitives in 1735 that are not represented by any samples in non-occluded layers of the buffer are considered to be newly occluded for the subimage. The index of these primitives in the stage I (which were transmitted to stage II along with each primitive) are transmitted in a retrograde fashion to stage I where the corresponding primitives are removed from 1750.

Detailed Description Text (185):
These transformed image-space primitives are passed to a second stage where they undergo rasterization in step 6140. In this rasterization step shading is not necessarily performed since the function of rasterization in this implementation is only to write to the buffer in a way that allows exposure areas and occluded primitives to be identified in later steps. Each image-space primitive transmitted to stage II is transmitted along with its corresponding index location in the first-stage local primitive display list, 6150. Primitives transmitted to stage II from stage I processors are stored in an image-space primitive list, 6135. Following rasterization of all primitives the subimage depth buffer, 6145, is processed in step 6155 to identify exposure regions by the methods previously described. Subsequently ray casting or approximate beam casting is performed in these exposure regions, also shown as part of step 6155, in the second stage to effect the visibility search for newly exposed primitives. This process identifies newly visible primitives which ate transformed and rasterized into the subimage depth buffer in step 6165. Primitives identified as newly visible are passed in a retrograde fashion to stage one to be added to the local primitive reprojection list 6150. An additional step is periodically conducted in stage II following step 6165. This step is the previously described read of the depth buffer, here 6145, to identify occluded primitives. This step employs the data structures of 6145 and 6135 which is the local image-space primitive list. As previously described, in this process primitives in 6135 that are not represented by any samples in non-occluded layers of the buffer are considered to be newly occluded for the subimage. The index of these primitives in the stage I (which were transmitted to stage II along with each primitive) are transmitted in a retrograde fashion to stage I where the corresponding primitives are removed from 6150.

Detailed Description Text (201):
In addition to determining the list of newly visible primitives, and the list of primitives that are newly invisible because of viewport outclipping, the server computes a list of primitives that have become occluded in the image stream during the frame interval. The combination of these three sources of visibility transition represents the visibility event information for the frame and is transmitted by the server to the client. The process of determining newly occluded primitives for a frame is periodically performed by the server. This determination need not be performed for every frame. Instead the frequency of this determination is selected to prevent accumulation of occluded primitives on the

client (destination) display list. This process is performed on the server by testing all newly occluded primitives for each subimage against the GVPA to determine which of the primitives newly occluded for a subimage are actually newly occluded for the entire image. The process of determining newly occluded primitives for a subimage as previously described is step 1570 of FIG. 15 for primitive reprojection. The newly occluded primitives for each subimage are determined, in the present parallel pipelined implementation, in the stage II subimage processor by the previously described process of reading non-occluded levels of the buffer after rasterization of all subimage primitives. The index of each newly occluded primitive in the stage I display list (previously transmitted with the image-space representation of the primitive) is transmitted to stage I to indicate newly occluded primitives for the subimage.

11. Document ID: US 5579454 A

L12: Entry 11 of 11 File: USPT Nov 26, 1996

DOCUMENT-IDENTIFIER: US 5579454 A
**** See image for Certificate of Correction ****
TITLE: Three dimensional graphics processing with pre-sorting of surface portions

Detailed Description Text (10):
The appearance of the object on a two dimensional screen such as the display unit 50, and consequently the image data stored in the frame buffer 40, is dependent upon the direction from which the object is to be viewed; it may also be dependent upon a defined distance between the object and a notional view point corresponding to the plane of the display unit 50. The computer 100 therefore includes a view control means 106 enabling an operator to define a view direction or direction and distance (for example, by defining the coordinates of a view point and/or a direction), via the input device 30. Rather than defining individual view points one at a time, the input means 30 may be employed to define a trajectory of successive view points or viewing directions, for example by specifying a direction of movement and a speed, so that successively rendered images are animated. The data defining the view direction and/or distance defines a relative distance and direction between the object and the notional plane of the display unit 50 (notional view point); where only a single object is to be displayed, it may allow data to define either the position and orientation of the object or of the screen, since what matters is the relative inclination between the two. On the other hand, where multiple objects at mutually defined positions are to be drawn, the view controller 106 preferably is capable of defining either the view position or a new position for one object at a time, so that either one object may be moved independently or the view may be moved relative to all objects.

Detailed Description Text (11):
Having defined the relative orientation of the object, relative to the viewing plane corresponding to the display unit 50, the computer 100 has the necessary data to enable projection means 108 to perform a projection of the or each three dimensional object (region by region) into the two dimensional viewing plane. Each two dimensional viewing plane region thus projected by the projector 108 is then rendered, in the order previously defined by the sorter 104, by a renderer 110 which fills the region (i.e., allocates a predetermined colour value to each pixel location in the frame buffer 40 which lies within the region). As will be discussed in greater detail below, only regions which face outwardly of the object ("forward facing") are rendered in preferred embodiments; the renderer 110 therefore is arranged to test each region to determine the direction which, in the projected plane, it is facing. Having rendered all regions and written the corresponding image data into the frame buffer 40, the view controller 106 updates the view point, if necessary, and, if it has changed, the projector 108 and renderer 110 re-execute their respective processes as above.

Detailed Description Text (39):
By referring back to FIG. 8a, it will be seen that in the case of a convex object, every polygon lies behind the plane of every other and no polygon can occlude any other. Referring to FIG. 11, however, and considering the polygons P1 and P3, whereas P1 lies in front of P3, P3 does not lie in front of P1. It is likewise apparent that, whereas P3 cannot occlude P1 from any angle in which both face forwards, P1 does occlude P3 from certain viewing angles. The condition that, considering a pair of polygons, each lies behind the plane of the other is therefore a test for

whether neither can occlude the other. The process of FIGS. 10 and 11 therefore effectively discriminates between potentially occluding polygons and non-occluding polygons, since after the execution of FIGS. 10 and 12, polygons which occlude no others will include no other polygons in their entries in the table 213.

Terms	Documents
L10 and I5	11